

# Local Gaussian Process Model for Large-scale Dynamic Computer Experiments

Ru Zhang, C. Devon Lin

Department of Mathematics and Statistics,  
Queen's University, ON, Canada

and

Pritam Ranjan

Operations Management & Quantitative Techniques,  
Indian Institute of Management Indore, MP, India

November 30, 2016

## Abstract

The recent accelerated growth in the computing power has generated popularization of experimentation with dynamic computer models in various physical and engineering applications. Despite the extensive statistical research in computer experiments, most of the focus had been on the theoretical and algorithmic innovations for the design and analysis of computer models with scalar responses.

In this paper, we propose a computationally efficient statistical emulator for a large-scale dynamic computer simulator (i.e., simulator which gives time series outputs). The main idea is to first find a good local neighbourhood for every input location, and then emulate the simulator output via a singular value decomposition (SVD) based Gaussian process (GP) model. We develop a new design criterion for sequentially finding this local neighbourhood set of training points. Several test functions and a real-life application have been used to demonstrate the performance of the proposed approach over a naive method of choosing local neighbourhood set using the Euclidean distance among design points.

*Keywords:* Nearest neighbour; Sequential design; Singular value decomposition; Statistical emulator; Time series output.

# 1. INTRODUCTION

Computer experiments are increasingly used in physical, engineering and social sciences as an economical alternative to physical experiments with complex systems/phenomena (Sacks et al., 1989; Santner et al., 2003). Such experiments are performed on computers with the underlying process represented and implemented by mathematical models. Although cheaper than physical experiments, realistic computer experiments for complex processes can still be time-consuming or sometimes infeasible, and thus, statistical surrogates or emulators are often used for thorough investigation.

Popular objectives of such computer experiments include estimation of pre-specified process features (e.g., overall response surface, global optimum, inverse problem, quantile, and so on), sensitivity analysis, calibration and uncertainty quantification (Jones et al. (1998); Kennedy and O’Hagan (2001); Ranjan et al. (2012); Bingham et al. (2014)). Despite the extensive statistical research in computer experiments, most of the focus had been on the theoretical and algorithmic innovations for the design and analysis of computer models with scalar responses. In this paper we focus on the emulation of dynamic computer models - referred to computer simulators with time series outputs.

Dynamic computer experiments arise in various applications, for example, rainfall-runoff model (Conti et al., 2009), and vehicle suspension system (Bayarri et al., 2007). Our motivating application comes from an apple farming industry where the objective is to emulate the population growth curve of European red mites which infest on apple leaves and diminish the crop quality (Teismann et al., 2009).

With the accelerated growth of computing power, and hence the availability of dynamic computer simulators, there is a desperate need for innovative methodologies and algorithms for the design and analysis of experiments that can particularly handle large data sets. In general, the size of data is a multiple of the length of the time series outputs. Recently, a few attempts have been made to emulate outputs of dynamic computer simulators. For instance, Kennedy and O’Hagan (2001) considered time as another input variable in the correlation structure and emulate the response via GP models. This has been the basis of several innovative ideas and methodologies since then (e.g., Stein (2005); Hung et al. (2015)). Conti et al. (2009) constructed dynamic emulators by using a one-step transition

function of state vectors to emulate the computer model movement from one time step to the next. Liu and West (2009) proposed time varying autoregression (TVAR) models with GP residuals. Another clever approach is to represent the time series outputs as linear combinations of a fixed set of basis such as singular vectors (Higdon et al., 2008) or wavelet basis (Bayarri et al., 2007) and impose GP models on the linear coefficients. However, fitting GP models over the entire training set can often be computationally infeasible for large-scale dynamic computer experiments involving thousands of training points.

We propose a new approach based on singular value decomposition (SVD) and the local surrogate idea, the latter of which was originally proposed for scalar valued computer simulators with large training data (Stein et al., 2004). The local surrogate idea was to emulate the process in a local neighbourhood of the input location of interest. Stein et al. (2004) used an Euclidean distance based nearest neighbour approach for finding the local neighbourhood. Gramacy and Apley (2015) further improved the prediction accuracy by using a sequential greedy algorithm and an optimality criterion for finding a non-trivial local neighbourhood set. Our objective is to generalize this optimality criterion for the sequential construction of the local neighbourhood set for emulating the dynamic computer simulators. We also develop an algorithm for implementation of the proposed methodology which is efficient from a large-data standpoint.

The subsequent sections are organized as follows. Section 2 reviews the concept of SVD-based GP models and provides a rigorous account for its model assumption and empirical Bayesian inference. Section 3 presents an innovative generalization of the optimality criterion, and a new algorithm for the greedy local approximate SVD-based GP models. We also compare the computational complexity of the algorithms. Section 4 uses several test functions to compare the performance of the naive (Euclidean distance based nearest neighbour) local SVD-based GP models, the full (using all training points) SVD-based GP models, and the proposed methodology in terms of prediction accuracy. The proposed method is also applied to the two-delay blowfly (TDB) model which simulates the population growth curve of European red mites. Concluding remarks are provided in Section 5, and proofs are given in the Appendix.

## 2. SVD-BASED GP MODELS

Higdon et al. (2008) proposed an SVD-based GP model for the calibration of computer simulators with highly multivariate outputs. They used a full Bayesian approach for model fitting which is exceedingly expensive for large-scale computer experiments, particularly in our proposed sequential procedure for fitting local SVD-based GPs. To address this computational issue, we propose an empirical Bayesian procedure. To this end, we first present a brief review of the SVD-based GP models proposed by Higdon et al. (2008), and then outline an empirical Bayesian procedure to reduce the computational burden.

### 2.1. MODEL FORMULATION

Consider a computer simulator which takes a  $q$ -dimensional quantitative input  $\mathbf{x} \in \mathbb{R}^q$ , and returns a time series output  $\mathbf{y}(\mathbf{x}) \in \mathbb{R}^L$  of length  $L$ .

For  $N$  training points, let  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]^T$  be the  $N \times q$  input matrix and  $\mathbf{Y} = [\mathbf{y}(\mathbf{x}_1), \dots, \mathbf{y}(\mathbf{x}_N)]$  be the  $L \times N$  matrix of time series responses. The SVD on  $\mathbf{Y}$  gives

$$\mathbf{Y} = \mathbf{U}\mathbf{D}\mathbf{V}^T,$$

where  $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_k]$  is an  $L \times k$  column-orthogonal matrix of left singular vectors, with  $k$  being the minimum of  $N$  and  $L$ ,  $\mathbf{D} = \text{diag}(d_1, \dots, d_k)$  is a  $k \times k$  diagonal matrix of singular values sorted in decreasing order, and the matrix  $\mathbf{V}$  is an  $N \times k$  column-orthogonal matrix of right singular vectors. The SVD-based GP model assumes that, for any  $\mathbf{x} \in \mathbb{R}^q$ ,

$$\mathbf{y}(\mathbf{x}) = \sum_{i=1}^p c_i(\mathbf{x})\mathbf{b}_i + \boldsymbol{\epsilon}(\mathbf{x}), \quad (1)$$

where the orthogonal basis  $\mathbf{b}_i = d_i\mathbf{u}_i \in \mathbb{R}^L$ , for  $i = 1, \dots, p$ , are the first  $p$  vectors of  $\mathbf{U}$  scaled by the corresponding singular values. For notational simplicity, we denote  $\mathbf{U}^* = [\mathbf{u}_1, \dots, \mathbf{u}_p]$ ,  $\mathbf{D}^* = \text{diag}(d_1, \dots, d_p)$ ,  $\mathbf{V}^* = [\mathbf{v}_1, \dots, \mathbf{v}_p]$  and  $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_p] = \mathbf{U}^*\mathbf{D}^*$ . The  $j$ th entry ( $1 \leq j \leq N$ ) of the  $N$ -dimensional vector  $\mathbf{v}_i$  ( $1 \leq i \leq p$ ) can be interpreted in two different ways. One can view it as the  $i$ th basis coefficient of the time series output  $\mathbf{y}(\mathbf{x}_j)$ , or alternatively, it can be treated as a realization of the Gaussian process  $c_i(\mathbf{x}_j)$ .

The number of significant singular values,  $p$  in (1), is determined empirically by the

cumulative percentage criterion

$$p = \min \left\{ m : \frac{\sum_{i=1}^m d_i}{\sum_{i=1}^k d_i} > \gamma \right\},$$

where  $\gamma$  is a prespecified threshold of explained variation (we used  $\gamma = 0.9$  as suggested by Jolliffe (2002)). The coefficients  $c_i$ 's in (1) are random functions (Rasmussen and Williams, 2006) assumed to be independent Gaussian processes, i.e.,  $c_i \sim \text{GP}(0, \sigma_i^2 K_i(\cdot, \cdot; \boldsymbol{\theta}_i))$  for  $i = 1, \dots, p$ . We use the popular anisotropic Gaussian correlation,

$$K(\mathbf{x}_1, \mathbf{x}_2; \boldsymbol{\theta}) = \exp \left\{ - \sum_{j=1}^q \theta_j (x_{1j} - x_{2j})^2 \right\},$$

for characterizing the spatial correlation structure, however, one can easily use another suitable correlation structure like Matérn or power-exponential (see Santner et al. (2003); Rasmussen and Williams (2006)). The random error  $\boldsymbol{\epsilon}(\mathbf{x})$  in (1) is assumed to be independent Gaussian white noise, that is,  $\boldsymbol{\epsilon}(\mathbf{x}) \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_L)$ .

Similar to Higdon et al. (2008), we use Bayesian algorithms for model fitting, however, since a full Bayesian implementation is too time consuming, we follow an empirical Bayesian approach. This is particularly crucial here as the GP models have to be fit several times in the proposed greedy sequential procedure.

## 2.2. EMPIRICAL BAYESIAN INFERENCE

This section briefly reviews the key components of our model fitting procedure. We follow the standard MCMC techniques like Metropolis Hasting and Gibbs algorithm for fitting the model described in Section 2.1, and use the maximum a posteriori (MAP) values as the plug-in estimates of the model parameters.

The parameters of interest are  $\sigma^2$  - the error variance, and for  $i = 1, 2, \dots, p$ , the process variance  $\sigma_i^2$  and the  $q$ -dimensional correlation hyper-parameter  $\boldsymbol{\theta}_i = (\theta_{i1}, \dots, \theta_{iq})$ . Similar to Gramacy and Apley (2015), we use inverse Gamma priors for  $\sigma_i^2$  and  $\sigma^2$ , i.e.,

$$[\sigma_i^2] \sim \text{IG} \left( \frac{\alpha_i}{2}, \frac{\beta_i}{2} \right), i = 1, \dots, p, \quad [\sigma^2] \sim \text{IG} \left( \frac{\alpha}{2}, \frac{\beta}{2} \right),$$

and use the mixture of Gamma prior for the hyper-parameter  $1/\theta_{ij}$  which captures both

the long-range and short-range dependencies. As a result, the posterior of  $\boldsymbol{\theta}_i$  becomes

$$\pi(\boldsymbol{\theta}_i|\mathbf{v}_i) \propto |\mathbf{K}_i|^{-\frac{1}{2}} \left( \frac{\beta_i + \psi_i}{2} \right)^{-(\alpha_i + N)/2} \pi(\boldsymbol{\theta}_i), \quad (2)$$

where  $\pi(\boldsymbol{\theta}_i)$  represents the prior of  $\boldsymbol{\theta}_i$ ,  $\mathbf{K}_i$  is the  $N \times N$  correlation matrix on the training set  $\mathbf{X}$  with the  $(j, k)$ th entry being  $K(\mathbf{x}_j, \mathbf{x}_k; \boldsymbol{\theta}_i)$ , for  $j, k = 1, \dots, N$ ,

$$\psi_i = \mathbf{v}_i^T \mathbf{K}_i^{-1} \mathbf{v}_i,$$

and  $\mathbf{v}_i$  is the  $i$ th column of  $\mathbf{V}^*$ .

It can also be shown that, for any input  $\mathbf{x}_0$ , the conditional distribution of  $c_i(\mathbf{x}_0)$  given  $(\mathbf{v}_i, \boldsymbol{\theta}_i)$  are independent non-central  $t$  with  $N - 1$  degrees of freedom, for  $i = 1, \dots, p$ , i.e.,

$$[c_i(\mathbf{x}_0)|\mathbf{v}_i, \boldsymbol{\theta}_i] \sim t_{N-1}(\hat{c}_i(\mathbf{x}_0|\mathbf{v}_i, \boldsymbol{\theta}_i), \hat{\sigma}_i^2(\mathbf{x}_0|\mathbf{v}_i, \boldsymbol{\theta}_i)),$$

where the location parameter is

$$\hat{c}_i(\mathbf{x}_0|\mathbf{v}_i, \boldsymbol{\theta}_i) = \mathbf{k}_i^T(\mathbf{x}_0) \mathbf{K}_i^{-1} \mathbf{v}_i,$$

with  $\mathbf{k}_i(\mathbf{x}_0) = [K(\mathbf{x}_0, \mathbf{x}_1; \boldsymbol{\theta}_i), \dots, K(\mathbf{x}_0, \mathbf{x}_N; \boldsymbol{\theta}_i)]^T$ , and the scale parameter is

$$\hat{\sigma}_i^2(\mathbf{x}_0|\mathbf{v}_i, \boldsymbol{\theta}_i) = \frac{(\beta_i + \psi_i) \left( 1 - \mathbf{k}_i^T(\mathbf{x}_0) \mathbf{K}_i^{-1} \mathbf{k}_i(\mathbf{x}_0) \right)}{\alpha_i + N - 1}.$$

Finally, the posterior distribution of  $\sigma^2$  given  $\mathbf{Y}$  is given by

$$\begin{aligned} \pi(\sigma^2|\mathbf{Y}) &\propto \pi(\mathbf{Y}|\sigma^2)\pi(\sigma^2) \\ &= (\sigma^2)^{-\frac{NL}{2}} \exp\left\{-\frac{\mathbf{r}^T \mathbf{r}}{2\sigma^2}\right\} (\sigma^2)^{-\frac{\alpha}{2}-1} \exp\left\{-\frac{\beta}{2\sigma^2}\right\} \\ &= (\sigma^2)^{-\frac{NL}{2}-\frac{\alpha}{2}-1} \exp\left\{-\frac{\mathbf{r}^T \mathbf{r} + \beta}{2\sigma^2}\right\}, \end{aligned} \quad (3)$$

where  $\pi(\sigma^2)$  is the prior distribution of  $\sigma^2$ , and  $\mathbf{r} = \text{vec}(\mathbf{Y}) - (\mathbf{I}_N \otimes \mathbf{B})\text{vec}(\mathbf{V}^{*T})$ , is the vectorization of residual matrix  $\mathbf{Y} - \mathbf{B}\mathbf{V}^{*T}$ . The notation  $\otimes$  represents the Kronecker product and the operator  $\text{vec}(\cdot)$  performs vectorization for a matrix. Thus,  $[\sigma^2|\mathbf{Y}]$  follows the inverse Gamma distribution  $\text{IG}((NL + \alpha)/2, (\mathbf{r}^T \mathbf{r} + \beta)/2)$ , and

$$\hat{\sigma}^2 = \underset{\sigma^2}{\text{argmax}} \pi(\sigma^2|\mathbf{Y}) = \frac{1}{NL + \alpha + 2} (\mathbf{r}^T \mathbf{r} + \beta). \quad (4)$$

The posterior predictive distribution of  $\mathbf{y}(\mathbf{x}_0)$  is given by

$$\pi(\mathbf{y}(\mathbf{x}_0)|\mathbf{V}^*, \boldsymbol{\Theta}, \sigma^2, \boldsymbol{\sigma}^2) \propto \int_{\mathbb{R}^p} \pi(\mathbf{y}(\mathbf{x}_0)|\mathbf{c}(\mathbf{x}_0), \sigma^2) \prod_{i=1}^p \pi(c_i(\mathbf{x}_0)|\mathbf{v}_i, \boldsymbol{\theta}_i) \prod_{i=1}^p dc_i(\mathbf{x}_0),$$

where  $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_p\}$ ,  $\mathbf{c}(\mathbf{x}_0) = (c_1(\mathbf{x}_0), \dots, c_p(\mathbf{x}_0))$  and  $\boldsymbol{\sigma}^2 = (\sigma_1^2, \dots, \sigma_p^2)$ . Note that  $\sigma_i^2$  is embedded in  $\pi(c_i(\mathbf{x}_0)|\mathbf{v}_i, \boldsymbol{\theta}_i)$ , and for a reasonably large value of  $N$ , a normal approximation can be imposed on the  $t_{N-1}$  distribution of  $[c_i(\mathbf{x}_0)|\mathbf{v}_i, \boldsymbol{\theta}_i]$ , i.e.,

$$\pi(c_i(\mathbf{x}_0)|\mathbf{v}_i, \boldsymbol{\theta}_i) \approx \mathcal{N}(\hat{c}_i(\mathbf{x}_0|\mathbf{v}_i, \boldsymbol{\theta}_i), \hat{\sigma}_i^2(\mathbf{x}_0|\mathbf{v}_i, \boldsymbol{\theta}_i)). \quad (5)$$

Furthermore, the results from Section 14.2 of Gelman et al. (2014) can be summarized into Lemma 1 for further simplification of the predictive distribution of  $\mathbf{y}(\mathbf{x}_0)$ .

**Lemma 1** Suppose  $[\mathbf{y}|\boldsymbol{\beta}, \sigma^2] \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I}_n)$  and  $[\boldsymbol{\beta}] \sim \mathcal{N}(\mathbf{b}, \mathbf{V})$ , where  $\mathbf{y} \in \mathbb{R}^n$ ,  $\boldsymbol{\beta}, \mathbf{b} \in \mathbb{R}^m$ ,  $\mathbf{X}$  is an  $n \times m$  matrix, and  $\mathbf{V}$  is an  $m \times m$  positive definite covariance matrix. Then,  $[\mathbf{y}|\sigma^2] \sim \mathcal{N}(\mathbf{X}\mathbf{b}, \mathbf{X}\mathbf{V}\mathbf{X}^T + \sigma^2 \mathbf{I}_n)$ .

Combining (1) and (5) with Lemma 1, we get

$$\pi(\mathbf{y}(\mathbf{x}_0)|\mathbf{V}^*, \boldsymbol{\Theta}, \sigma^2, \boldsymbol{\sigma}^2) \approx \mathcal{N}(\mathbf{B}\hat{\mathbf{c}}(\mathbf{x}_0|\mathbf{V}^*, \boldsymbol{\Theta}), \mathbf{B}\boldsymbol{\Lambda}(\mathbf{V}^*, \boldsymbol{\Theta})\mathbf{B}^T + \sigma^2 \mathbf{I}_L), \quad (6)$$

where  $\hat{\mathbf{c}}(\mathbf{x}_0|\mathbf{V}^*, \boldsymbol{\Theta}) = [\hat{c}_1(\mathbf{x}_0|\mathbf{v}_1, \boldsymbol{\theta}_1), \dots, \hat{c}_p(\mathbf{x}_0|\mathbf{v}_p, \boldsymbol{\theta}_p)]^T$ , and  $\boldsymbol{\Lambda}(\mathbf{V}^*, \boldsymbol{\Theta}) = \text{diag}(\hat{\sigma}_1^2(\mathbf{x}_0|\mathbf{v}_1, \boldsymbol{\theta}_1), \dots, \hat{\sigma}_p^2(\mathbf{x}_0|\mathbf{v}_p, \boldsymbol{\theta}_p))$ . The parameters  $\boldsymbol{\Theta}$  and  $\sigma^2$  cannot be integrated out analytically, and Kennedy and O'Hagan (2001) suggested using the MAP estimator into the predictive distribution. Following their paradigm, we plug  $\hat{\sigma}^2$  and

$$\hat{\boldsymbol{\theta}}_i = \underset{\boldsymbol{\theta}_i}{\operatorname{argmax}} \pi(\boldsymbol{\theta}_i|\mathbf{v}_i), \quad i = 1, \dots, p, \quad (7)$$

into (6) to obtain the approximate predictive distribution

$$\pi(\mathbf{y}(\mathbf{x}_0)|\mathbf{Y}) \approx \mathcal{N}(\mathbf{B}\hat{\mathbf{c}}(\mathbf{x}_0|\mathbf{V}^*, \hat{\boldsymbol{\Theta}}), \mathbf{B}\boldsymbol{\Lambda}(\mathbf{V}^*, \hat{\boldsymbol{\Theta}})\mathbf{B}^T + \hat{\sigma}^2 \mathbf{I}_L), \quad (8)$$

where  $\pi(\boldsymbol{\theta}_i|\mathbf{v}_i)$  and  $\pi(\sigma^2|\mathbf{Y})$  are given by (2) and (3), respectively. The response matrix  $\mathbf{Y}$  is used in (8) instead of  $\mathbf{V}^*$  since  $\mathbf{V}^*$  is determined by  $\mathbf{Y}$  through SVD.

The SVD-based GP model fitting via the empirical Bayesian procedure outlined thus far is summarized in Algorithm 1. The predictive distribution stated in (8) can be obtained by plugging the output of Algorithm 1.

---

**Algorithm 1:** Empirical Bayesian inference for SVD-based GP models

---

**Input** : (1) Training set:  $\mathbf{X}_{N \times q}$ , (2) response matrix:  $\mathbf{Y}_{L \times N}$ , (3) threshold  $\gamma$ ,  
(4) prior parameters:  $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_p]^T$ ,  $\boldsymbol{\beta} = [\beta_1, \dots, \beta_p, \beta]^T$ .

**Output:** (1) Basis matrix  $\mathbf{B}_{N \times p}$ , (2) singular value matrix  $\mathbf{D}_{p \times p}^*$ , (3) coefficient matrix  $\mathbf{V}^*$ , (4) correlation parameters  $\hat{\boldsymbol{\Theta}}$ , (5) noise variances  $\hat{\sigma}^2$  and  $\hat{\boldsymbol{\sigma}}^2 = (\hat{\sigma}_1^2, \dots, \hat{\sigma}_p^2)$ .

---

```
1 Function svdGP( $\mathbf{X}, \mathbf{Y}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma$ )
2   [ $\mathbf{B}, \mathbf{D}^*, \mathbf{V}^*, p$ ]  $\leftarrow$  buildBasis( $\mathbf{Y}, \gamma$ )
3    $\mathbf{r} \leftarrow \text{vec}(\mathbf{Y}) - (\mathbf{I}_N \otimes \mathbf{B})\text{vec}(\mathbf{V}^{*T})$ 
4    $\hat{\sigma}_i^2(\mathbf{x}_0 | \mathbf{v}_i, \boldsymbol{\theta}_i) = (\beta_i + \psi_i) \left( 1 - \mathbf{k}_i^T(\mathbf{x}_0) \mathbf{K}_i^{-1} \mathbf{k}_i(\mathbf{x}_0) \right) / (\alpha_i + N - 1)$ ,          /* where
       $\psi_i = \mathbf{v}_i^T \mathbf{K}_i^{-1} \mathbf{v}_i$ , with  $\mathbf{v}_i$  as the  $i$ -th column of  $\mathbf{V}^*$ ,  $\mathbf{K}_i = K(\cdot, \cdot; \boldsymbol{\theta}_i)$  */
5    $\hat{\sigma}^2 \leftarrow (\mathbf{r}^T \mathbf{r} + \beta) / (NL + \alpha + 2)$ 
6    $\hat{\boldsymbol{\Theta}} \leftarrow \text{inference}(\mathbf{V}^*, p, \boldsymbol{\alpha}, \boldsymbol{\beta})$ 
7   return  $\mathbf{B}, \mathbf{D}^*, \mathbf{V}^*, \hat{\boldsymbol{\Theta}}, \hat{\sigma}^2, \hat{\boldsymbol{\sigma}}^2$ 

8 Subroutine buildBasis( $\mathbf{Y}, \gamma$ )
9   [ $\mathbf{U}, \mathbf{D}, \mathbf{V}$ ]  $\leftarrow$  SVD( $\mathbf{Y}$ )                                /* perform SVD on matrix  $\mathbf{Y}$ . */
10   $p \leftarrow \min \left\{ m : \frac{\sum_{i=1}^m d_i}{\sum_{i=1}^k d_i} > \gamma \right\}$  /* where  $\mathbf{D} = \text{diag}(d_1, \dots, d_N)$ ,  $k = \min\{N, L\}$  */
11   $\mathbf{B} \leftarrow \mathbf{U}^* \mathbf{D}^*$                                           /* as in Section 2.1 */
12  return  $\mathbf{B}, \mathbf{D}^*, \mathbf{V}^*, p$ 

13 Subroutine inference( $\mathbf{V}^*, p, \boldsymbol{\alpha}, \boldsymbol{\beta}$ )
14   /* fit  $p$  independent GPs by finding the MAPs */
15   for  $i \leftarrow 1$  to  $p$  do
16      $\hat{\boldsymbol{\theta}}_i \leftarrow \underset{\boldsymbol{\theta}_i}{\text{argmax}} \pi(\boldsymbol{\theta}_i | \mathbf{v}_i)$           /*  $\boldsymbol{\theta}_i$  is  $q$ -dimensional, also see (2) */
17    $\hat{\boldsymbol{\Theta}} \leftarrow [\hat{\boldsymbol{\theta}}_1, \dots, \hat{\boldsymbol{\theta}}_p]^T$ 
18   return  $\hat{\boldsymbol{\Theta}}$ 
```

---

Fitting the  $i$ -th GP model ( $1 \leq i \leq p$ ) to  $N$  training data points (Line 14, Algorithm 1) involves numerous evaluations of the posterior (2), and the computation of  $\mathbf{K}_i^{-1}$  and  $|\mathbf{K}_i|$



requires  $O(N^3)$  floating point operations (flops), which can quickly become infeasible even for moderately large  $N$ . Thus, we propose to use a localized SVD-based GP model that aims to achieve the same prediction accuracy at a substantially less computational cost.

### 3. LOCAL SVD-BASED GP MODEL

The main idea is to use a small subset of  $n$  ( $\ll N$ ) points instead of the entire training set of  $N$  points for approximating the predicted response at an arbitrary  $\mathbf{x}_0$  in the input space. Let  $\mathbf{X}$  be the training set of  $N$  points, and  $\mathbf{X}^{(n)}(\mathbf{x}_0)$  or  $\mathbf{X}^{(n)}$  (in short) denote the desired subset of  $\mathbf{X}$  which defines the neighbourhood of  $\mathbf{x}_0$  contained in  $\mathbf{X}$ . In this section, we discuss two methods of constructing this neighbourhood set  $\mathbf{X}^{(n)}$ .

The first one, called as the *naive* approach, assumes the elements of the neighbourhood set  $\mathbf{X}^{(n)}$  by finding  $n$  nearest neighbours of  $\mathbf{x}_0$  in  $\mathbf{X}$  as per the Euclidean distance. The emulator obtained via fitting an SVD-based GP model (as described in Section 2) to this local set of points is referred to as *naive local SVD-based GP model* (in short, nlGP). Though, nlGP is computationally much cheaper than the *full SVD-based GP model* (fGP) trained on  $N$  points, its prediction accuracy may not satisfactory.

The second method (main focus of this paper) finds the neighbourhood set  $\mathbf{X}^{(n)}(\mathbf{x}_0)$  (for every  $\mathbf{x}_0$ ) using a greedy approach. Gramacy and Apley (2015) developed a greedy sequential algorithm for constructing a neighbourhood set for a scalar-valued simulator. In this paper, we propose a generalization of this algorithm for dynamic computer simulators. The generalized greedy algorithm finds a local set of points which are used to build an SVD-based GP model. Such a model is referred to as *greedy local SVD-based GP model* (in short, glGP).

#### 3.1. GREEDY LOCAL SVD-BASED GP MODEL

The greedy approach starts with finding a smaller neighbourhood set  $\mathbf{X}^{(n_0)}(\mathbf{x}_0)$ , which consists of  $n_0$  ( $< n$ ) nearest neighbours of  $\mathbf{x}_0$  in  $\mathbf{X}$  (with respect to the Euclidean distance). This step is the same as in nlGP with  $n$  replaced by  $n_0$ . For every  $\mathbf{x}_0$  in the input space (or the test set), the remaining  $n - n_0$  neighbourhood points are chosen sequentially with the aim of reducing the prediction error.

Let  $k$  denote the current number of points in the neighbourhood set,  $\mathbf{X}^{(k)}$  and  $\mathbf{X} \setminus \mathbf{X}^{(k)}$  be the sets of selected and unselected (remaining) training points, respectively, and  $\hat{\boldsymbol{\Theta}}^{(k)} = \{\hat{\boldsymbol{\theta}}_1^{(k)}, \dots, \hat{\boldsymbol{\theta}}_p^{(k)}\}$  be the estimated correlation hyper-parameters obtained using  $\text{svdGP}(\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$  of Algorithm 1. Then the next follow-up point is chosen as

$$\mathbf{x}_{k+1}^* = \underset{\mathbf{x} \in \mathbf{X} \setminus \mathbf{X}^{(k)}}{\operatorname{argmin}} J(\mathbf{x}_0, \mathbf{x}),$$

where

$$J(\mathbf{x}_0, \mathbf{x}) = \mathbb{E} \left\{ \mathbb{E} \left[ \left\| \mathbf{y}(\mathbf{x}_0) - \hat{\mathbf{y}}(\mathbf{x}_0 | \mathbf{c}(\mathbf{x}), \mathbf{V}^{*(k)}, \hat{\boldsymbol{\Theta}}^{(k)}) \right\|^2 \middle| \mathbf{c}(\mathbf{x}), \mathbf{V}^{*(k)}, \hat{\boldsymbol{\Theta}}^{(k)}, (\hat{\sigma}^{(k)})^2 \right] \middle| \mathbf{V}^{*(k)}, \hat{\boldsymbol{\Theta}}^{(k)}, (\hat{\sigma}^{(k)})^2 \right\}, \quad (9)$$

with

$$\begin{aligned} \hat{\mathbf{y}}(\mathbf{x}_0 | \mathbf{c}(\mathbf{x}), \mathbf{V}^{*(k)}, \hat{\boldsymbol{\Theta}}^{(k)}) &= \mathbb{E} \left[ \mathbf{y}(\mathbf{x}_0) \middle| \mathbf{c}(\mathbf{x}), \mathbf{V}^{*(k)}, \hat{\boldsymbol{\Theta}}^{(k)}, (\hat{\sigma}^{(k)})^2 \right] \\ &= \mathbf{B}^{(k)} \hat{\mathbf{c}}(\mathbf{x}_0 | \mathbf{c}(\mathbf{x}), \mathbf{V}^{*(k)}, \hat{\boldsymbol{\Theta}}^{(k)}), \end{aligned} \quad (10)$$

where  $\mathbf{B}^{(k)}$  and  $\mathbf{V}^{*(k)}$  are the matrices of basis vectors and the right singular vectors returned by  $\text{buildBasis}(\mathbf{Y}^{(n)}, \gamma)$  in Algorithm 1,  $p_k$  is the number of bases selected in this iteration, and  $\mathbf{c}(\mathbf{x}) = [c_1(\mathbf{x}), \dots, c_{p_k}(\mathbf{x})]^T$  with  $c_i \sim \text{GP}(0, K(\cdot, \cdot; \hat{\boldsymbol{\theta}}_i^{(k)}))$ . The predictive mean vector of coefficients  $\hat{\mathbf{c}}(\mathbf{x}_0 | \mathbf{c}(\mathbf{x}), \mathbf{V}^{*(k)}, \hat{\boldsymbol{\Theta}}^{(k)})$  is calculated in the exact same way as (6) except  $[(\mathbf{V}^{*(k)})^T, \mathbf{c}(\mathbf{x})]^T$  and  $\boldsymbol{\Theta}^{(k)}$  are used in place of  $\mathbf{V}^*$  and  $\boldsymbol{\Theta}$ , respectively.

The  $J(\cdot, \cdot)$  is equivalent to the mean squared prediction generalization error. This  $J$ -criterion is inspired by the active learning Cohn (ALC) criterion (Cohn et al., 1996)

$$\int_{\mathbf{x}} \left[ \int_y (\hat{y}(\mathbf{x}) - y(\mathbf{x}))^2 dP(y|\mathbf{x}) \right] dP(\mathbf{x}),$$

where  $y(\mathbf{x})$  and  $\hat{y}(\mathbf{x})$  are the observed and predicted scalar-valued outputs, respectively, at input  $\mathbf{x}$ ,  $P(y|\mathbf{x})$  is the approximate predictive distribution, and the marginal distribution  $P(\mathbf{x})$  is uniform. For dynamic computer simulators, we use  $L_2$  norm discrepancy instead of the squared error.

The closed form expression for  $J(\mathbf{x}_0, \mathbf{x})$  can be derived by taking the outer expectation in (9) with respect to the approximate posterior distribution in (6) and substituting  $(\mathbf{V}^{*(k)}, \hat{\boldsymbol{\Theta}}^{(k)}, (\hat{\sigma}^{(k)})^2)$  for  $(\mathbf{V}^*, \boldsymbol{\Theta}, \sigma^2)$ . Similarly, the expectation in (10) is computed with respect to (6), and by substituting  $[(\mathbf{V}^{*(k)})^T, \mathbf{c}(\mathbf{x})]^T, \hat{\boldsymbol{\Theta}}^{(k)}, (\hat{\sigma}^{(k)})^2$  for  $(\mathbf{V}^*, \boldsymbol{\Theta}, \sigma^2)$ . Proposition 1 states the closed form expression of  $J(\mathbf{x}_0, \mathbf{x})$ , and the proof is shown in the Appendix.

**Proposition 1** Suppose the expectations in (9) and (10) are taken with respect to the approximate predictive distribution (6). Then, for any  $\mathbf{x} \in \mathbf{X} \setminus \mathbf{X}^{(k)}$

$$J(\mathbf{x}_0, \mathbf{x}) = (\hat{\sigma}^{(k)})^2 L + \sum_{i=1}^{p_k} (d_i^{(k)})^2 \hat{\sigma}_i^2(\mathbf{x}_0 | \mathbf{x}, \mathbf{v}_i^{(k)}, \hat{\boldsymbol{\theta}}_i^{(k)}),$$

where  $d_i^{(k)}$  is the  $i$ -th largest singular value of  $\mathbf{Y}^{(k)}$ ,

$$\begin{aligned} \hat{\sigma}_i^2(\mathbf{x}_0 | \mathbf{x}, \mathbf{v}_i^{(k)}, \hat{\boldsymbol{\theta}}_i^{(k)}) &= \frac{\rho_i^{(k)}(\mathbf{x}_0, \mathbf{x})}{\alpha_i + k} \left( \beta_i + \frac{\alpha_i + k}{\alpha_i + k - 1} \psi_i^{(k)} \right), \\ \rho_i^{(k)}(\mathbf{x}_0, \mathbf{x}) &= 1 - \tilde{\mathbf{k}}_i(\mathbf{x}_0, \mathbf{x})^T \tilde{\mathbf{K}}_i^{-1}(\mathbf{x}) \tilde{\mathbf{k}}_i(\mathbf{x}_0, \mathbf{x}), \\ \psi_i^{(k)} &= (\mathbf{v}_i^{(k)})^T (\mathbf{K}_i^{(k)})^{-1} \mathbf{v}_i^{(k)}, \\ \tilde{\mathbf{k}}_i(\mathbf{x}_0, \mathbf{x}) &= [K(\mathbf{x}_0, \mathbf{x}_1^{(k)}; \hat{\boldsymbol{\theta}}_i^{(k)}), \dots, K(\mathbf{x}_0, \mathbf{x}_k^{(k)}; \hat{\boldsymbol{\theta}}_i^{(k)}), K(\mathbf{x}_0, \mathbf{x}; \hat{\boldsymbol{\theta}}_i^{(k)})]^T, \\ \tilde{\mathbf{K}}_i(\mathbf{x}) &= \begin{bmatrix} \mathbf{K}_i^{(k)} & \mathbf{k}_i^{(k)}(\mathbf{x}) \\ \mathbf{k}_i^{(k)}(\mathbf{x})^T & 1 \end{bmatrix}, \end{aligned}$$

with  $\mathbf{v}_i^{(k)}$  being the  $i$ -th column of  $\mathbf{V}^{*(k)}$ , for  $i = 1, \dots, p_k$ ,  $\mathbf{x}_j^{(k)}$  being the  $j$ -th point of  $\mathbf{X}^{(k)}$  for  $j = 1, \dots, k$ ,  $\mathbf{K}_i^{(k)}$  being a  $k \times k$  matrix with  $K(\mathbf{x}_j^{(k)}, \mathbf{x}_l^{(k)}; \hat{\boldsymbol{\theta}}_i^{(k)})$ , as the  $(j, l)$ -th entry, and  $\mathbf{k}_i^{(k)}(\mathbf{x}) = [K(\mathbf{x}, \mathbf{x}_1^{(k)}; \hat{\boldsymbol{\theta}}_i^{(k)}), \dots, K(\mathbf{x}, \mathbf{x}_k^{(k)}; \hat{\boldsymbol{\theta}}_i^{(k)})]^T$ .

As  $(\hat{\sigma}^{(k)})^2 L$  is a constant with respect to  $\mathbf{x}$ , finding  $\mathbf{x}_{k+1}^*$  by minimizing the  $J$ -criterion in Proposition 1 is equivalent to obtaining

$$\mathbf{x}_{k+1}^* = \underset{\mathbf{x} \in \mathbf{X} \setminus \mathbf{X}^{(k)}}{\operatorname{argmin}} \left[ \sum_{i=1}^{p_k} (d_i^{(k)})^2 \hat{\sigma}_i^2(\mathbf{x}_0 | \mathbf{x}, \mathbf{v}_i^{(k)}, \hat{\boldsymbol{\theta}}_i^{(k)}) \right]. \quad (11)$$

Note that the simplified criterion in (11) turns out to be the weighted sum of the predictive variance of the singular vector coefficients, where the weights are  $(d_i^{(k)})^2$  which represents the total variation explained by the  $i$ -th singular vector basis. Therefore, the follow-up point  $\mathbf{x}_{k+1}^*$  minimizes the expected  $L_2$  prediction error at  $\mathbf{x}_0$  evaluated at stage  $k$ .

As compared to nlGP, the proposed algorithm, glGP, requires many more GP model fitting steps, which increases the computational cost, however, it is still substantially faster than the fGP implementation. To further reduce the computational burden, we apply the matrix inverse updating procedure based on Hager (1989) (also used in Gramacy and Apley (2015)), to achieve time saving from  $O(k^3)$  to  $O(k^2)$ . Note that the anticipated

boost in the prediction accuracy at the cost of a small increase in the computational cost is perhaps worth it. Algorithm 2 summarizes the glGP implementation, and the computational complexities of the two algorithms nlGP and glGP are extensively discussed in Section 3.2.

---

**Algorithm 2:** Greedy local SVD-based GP model

---

**Input** : (1) Training set:  $\mathbf{X}_{N \times q}$ , (2) response matrix:  $\mathbf{Y}_{L \times N}$ , (3) test set  $\mathbf{X}_{M \times q}^*$ ,  
(4) neighborhood size  $n$ , (5) initial neighborhood size  $n_0$ , (6) threshold  $\gamma$ ,  
(7) prior parameters  $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_p, \alpha]^T$  and  $\boldsymbol{\beta} = [\beta_1, \dots, \beta_p, \beta]^T$ .

**Output:** (1) The predicted mean response, and (2) the associated posterior variance in estimating  $\mathbf{y}(\mathbf{x}_0)$  for each  $\mathbf{x}_0 \in \mathbf{X}^*$ .

---

```

1 for each  $\mathbf{x}_0 \in \mathbf{X}^*$  do
2    $\mathbf{X}^{(n_0)} \leftarrow \{\mathbf{x}_i, i = 1, \dots, n_0\}$  /*  $n_0$  nearest neighbours of  $\mathbf{x}_0$  in  $\mathbf{X}$  as in nlGP */
3    $\mathbf{Y}^{(n_0)} \leftarrow \{y(\mathbf{x}) : \mathbf{x} \in \mathbf{X}^{(n_0)}\}$ 
4   for  $k \leftarrow n_0$  to  $n - 1$  do
5      $[\mathbf{B}^{(k)}, \mathbf{D}^{*(k)}, \mathbf{V}^{*(k)}, p_k, \hat{\boldsymbol{\Theta}}^k, (\hat{\sigma}^{(k)})^2, (\hat{\boldsymbol{\sigma}}^{(k)})^2] \leftarrow \text{svdGP}(\mathbf{X}^{(k)}, \mathbf{Y}^{(k)}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ 
6      $\mathbf{x}_{k+1}^* \leftarrow \underset{\mathbf{x} \in \mathbf{X} \setminus \mathbf{X}^{(k)}}{\text{argmin}} J(\mathbf{x}_0, \mathbf{x})$ 
7      $\mathbf{X}^{(k+1)} \leftarrow \mathbf{X}^{(k)} \cup \mathbf{x}_{k+1}^*$ 
8      $\mathbf{Y}^{(k+1)} \leftarrow \mathbf{Y}^{(k)} \cup \mathbf{y}(\mathbf{x}_{k+1}^*)$ 
9    $[\mathbf{B}^{(n)}, \mathbf{D}^{*(n)}, \mathbf{V}^{*(n)}, p_n, \hat{\boldsymbol{\Theta}}^{(n)}, (\hat{\sigma}^{(n)})^2, (\hat{\boldsymbol{\sigma}}^{(n)})^2] \leftarrow \text{svdGP}(\mathbf{X}^{(n)}, \mathbf{Y}^{(n)}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ 
10  Predict  $\mathbf{y}(\mathbf{x}_0)$  through  $\pi(\mathbf{y}(\mathbf{x}_0) | \mathbf{V}^{*(n)}, \hat{\boldsymbol{\Theta}}^{(n)}, (\hat{\sigma}^{(n)})^2, (\hat{\boldsymbol{\sigma}}^{(n)})^2)$  in (6)
```

---

### 3.2. COMPUTATIONAL COMPLEXITY

In this section, we discuss the computational complexity of (1) full SVD-based GP model (fGP), (2) naive local SVD-based GP model (nlGP), and (3) greedy local SVD-based GP model (glGP). For this comparison, let  $\mathbf{X}$  contain  $N$  training points,  $\mathbf{X}^*$  consist of  $M$  test points,  $L$  be the length of time series, each neighborhood set in nlGP and glGP consist of  $n$  training points, and  $N > L > n$  (assuming  $N$  is large). Furthermore, we only compute the

diagonal entries of the predictive covariance matrix of  $\mathbf{y}(\mathbf{x}_0)$ , i.e., the marginal predictive variances, for each  $\mathbf{x}_0 \in \mathbf{X}^*$ .

(1) **fGP**: A single call of **svdGP** on the full training data requires  $O(N^3)$  floating point operations (flops). Since  $N > L$  is assumed, the estimation of  $\Theta$  is the dominant part of **svdGP** computation, which requires  $O(N^3)$  flops. The complexity in the prediction step is  $O(M(N^2 + L)) = O(MN^2)$ , and thus, the total cost of fGP is  $O(N^2 \max\{M, N\})$ .

(2) **nlGP**: For each  $\mathbf{x}_0 \in \mathbf{X}^*$ , the neighbourhood set construction needs  $O(nN)$  flops, one call of **SVD** on line 8 of Algorithm 1 takes  $O(nL \min\{n, L\})$  flops (Gentle, 2007), which is  $O(n^2L)$  since  $n < L$  is assumed. The estimation of  $(\hat{\sigma}^{(n)})^2$  and  $\hat{\Theta}^{(n)}$  on lines 5 and 14 requires  $O(nL)$  and  $O(n^3)$  flops, respectively. That is, the cost of **svdGP** is  $O(n^2L + nL + n^3) = O(n^2L)$ . Furthermore, the computational complexity in the prediction step is  $O(n^2 + L)$ . Consequently, the total cost of nlGP is  $O(Mn \max\{nL, N\})$ .

(3) **glGP**: The cost of **svdGP** on line 5 of Algorithm 2 is  $O(k^2L)$ , as in nlGP. Optimization of the  $J$ -criterion costs  $O(k^2N)$  (as per the quick update formula by Gramacy and Apley (2015)). Thus the loop for the  $k$ -th iteration costs  $O(k^2N)$ , and the entire loop for  $k = n_0, \dots, n - 1$  requires  $\sum_{k=n_0}^{n-1} O(k^2N) = O(n^3N)$  flops. Note that the neighbourhood construction in line 2 and the prediction cost in line 10 are not significant as compared to the other cost. As a result, the total cost of this algorithm is  $O(n^3NM)$ .

Table 1 summarizes the computational complexity in the three algorithms. It is easy to see that glGP is computationally more expensive than nlGP, however, the gain in the prediction accuracy is perhaps worth more. Assuming  $M = O(N)$ , it is also straightforward to notice that nlGP and glGP are substantially faster than fGP as long as  $n = o(N^{1/3})$ .

Table 1: The computational cost of fitting GP models under the three algorithms.

Method	fGP	nlGP	glGP
Cost	$O(N^2 \max\{M, N\})$	$O(Mn \max\{nL, N\})$	$O(n^3NM)$

### 3.3. IMPLEMENTATION

Both the local SVD-based GP models can be run in a parallel computing environment. One quick option is to divide the job into  $M$  parts and fit independent local GP models. In

contrast, fGP models cannot be parallelized in such an easy manner, except the prediction component. Of course, one could use parallelization for SVD of  $\mathbf{Y}$ , and/or computing the determinant and inverse of the correlation matrices within the optimization step.

We implemented the three algorithms in R (R Core Team, 2015). The parallelization of the empirical Bayesian estimation (the *for*-loop on line 14 of Algorithm 1) and the outer loop of Algorithm 2 are implemented via the package *parallel* (R Core Team, 2015). The optimization in empirical Bayesian inference for all the three methods is performed with the assistance of the *laGP* package (Gramacy, 2015). It turned out that *laGP* gave unreliable fits for finding MAP estimators of  $\boldsymbol{\theta}$  under the fGP models, and thus alternatively we used *mlegp* package (Dancik, 2013) which is much slower than *laGP*.

## 4. Applications

In this section, we consider several examples with different test functions that represent dynamic computer models. We also consider a real-life application where the computer simulator (TDB model) generates population growth curve. The examples considered here range from  $N = 60$  to 10,000 (size of the training set), and  $q = 2$  to 11 (input dimension).

The performance of the three methods fGP, nlGP and glGP is evaluated by comparing the normalized mean squared prediction error (NMSPE),

$$\text{NMSPE}(\mathbf{x}) = \frac{\sum_{t=1}^L (y_t(\mathbf{x}) - \hat{y}_t(\mathbf{x}))^2}{\sum_{t=1}^L (y_t(\mathbf{x}) - \bar{y}(\mathbf{x}))^2}, \quad (12)$$

where  $y_t(\mathbf{x})$  is the real (scalar) response for input  $\mathbf{x}$  at time  $t$ ,  $t = 1, \dots, L$ ,  $\hat{y}_t(\mathbf{x})$  is the corresponding model prediction, and the temporal mean is given by  $\bar{y}(\mathbf{x}) = \sum_{t=1}^L y_t(\mathbf{x})/L$ . The normalization is applied for consistency of results among examples.

For all these methods, we use the vague scale-invariant priors (Gramacy, 2005) with  $\alpha_i$ 's,  $\beta_i$ 's,  $\alpha$  and  $\beta$  set to be 0, and for the correlation parameters  $\boldsymbol{\theta}_i$ 's, the flat priors are used. For the simulated test functions, Examples 1 to 3, we repeat the emulation procedure 60 times with different training and test data sets and compare the average performance.

#### 4.1. EXAMPLE 1 (Gramacy and Lee, 2012)

Suppose the simulator output is generated from the following test function (obtained by a slight modification of a test function in Gramacy and Lee (2012))

$$f(\mathbf{x}, t) = \frac{\sin(x_1 \pi t)}{x_2 t} + (t - 1)^4, \quad (13)$$

where  $\mathbf{x} = (x_1, x_2)^T \in [6, 14] \times [1, 3]$ , and  $t \in [0.5, 2.5]$  is on a 200-point equidistant time-grid. Note that the input  $x_1$  controls the frequency and  $x_2$  controls the amplitude. Thus, the outputs at different inputs attenuate sine waves with different periods.

For each of 60 replications, both the training and test data were randomly generated using a 500-point maximin Latin hypercube design (LHD) (Morris and Mitchell, 1995) from the input space  $[6, 14] \times [1, 3]$ . The local approximate methods are performed on the neighborhood set consisting of  $n = 30, 50$  points. For glGP, we assumed the initial neighborhood size to be  $n_0 = \lceil n/4 \rceil, \lceil n/2 \rceil$ , where  $\lceil x \rceil$  represents the smallest integer greater than or equal to  $x$ . Figure 1 summarizes  $\log(\text{NMSPE})$  values for different models. Notation:  $Nn$  represents naive method (nlGP) with neighbourhood set size  $n$ , and  $Gn/n_0$  denotes greedy method (glGP) with  $n$ -point neighbourhood among which  $n_0$  points are chosen using an LHD and the remaining  $n - n_0$  points are chosen sequentially by optimising the  $J$ -criterion.

It is clear from Figure 1 that glGP with  $n = 50$  and  $n_0 = \lceil n/2 \rceil$  performs the best among the settings we tried. Note that choosing  $n_0$  very small is not necessarily the best strategy in such sequential design approaches. Though the full GP implementation is computationally expensive, it is expected to be at least as accurate as the local GP models. Surprisingly, fGP implementations could not outperform any of the local GP models. As suspected, mlegp implementation of fGP gives more accurate prediction than that of laGP.

#### 4.2. EXAMPLE 2 (Forrester et al., 2008)

Consider the following test function with 3-dimensional inputs to generate simulator responses with time-series outputs,

$$f(\mathbf{x}, t) = (x_1 t - 2)^2 \sin(x_2 t - x_3), \quad (14)$$

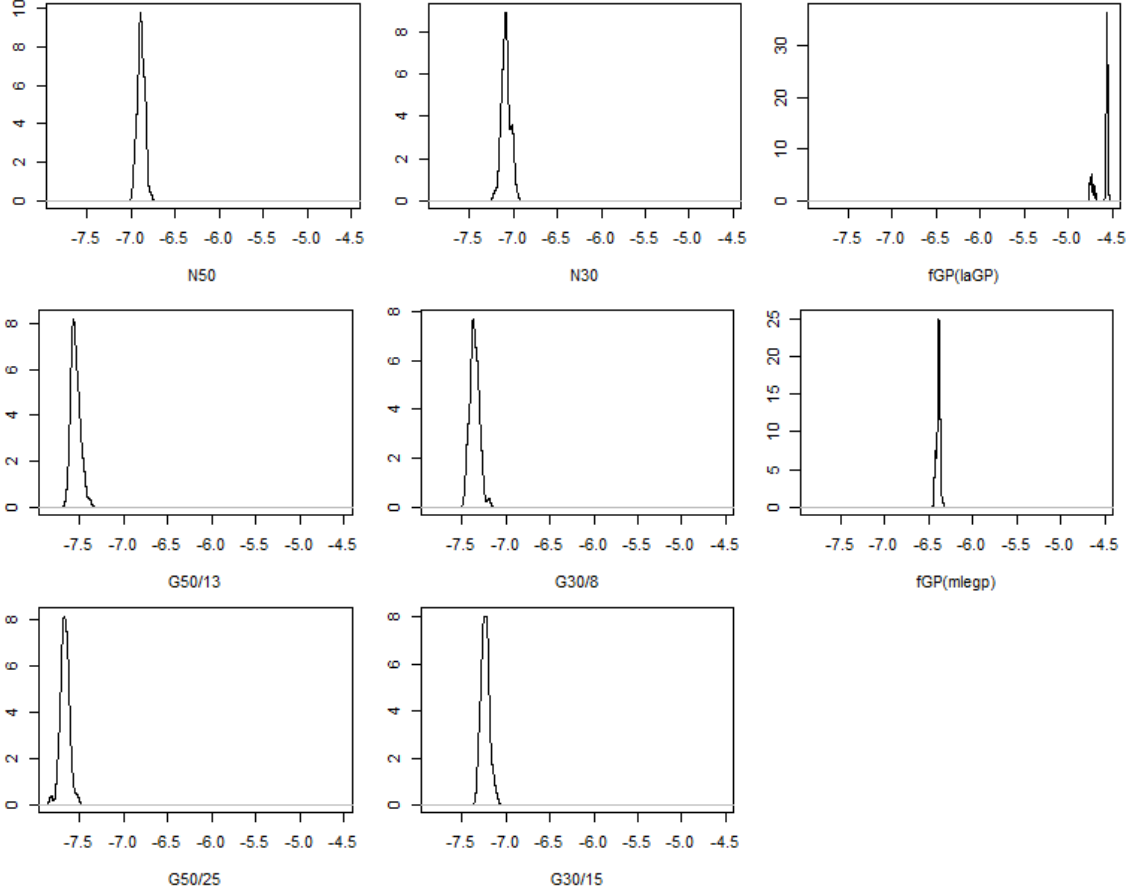


Figure 1: Density plots of the mean  $\log(\text{NMSPE})$  computed from 500 test points over 60 simulations for the computer simulator (13). The leftmost panel uses  $n = 50$ , the middle one uses  $n = 30$  and the rightmost column represents the full 500-point design.

where  $\mathbf{x} = (x_1, x_2, x_3)^T \in [4, 10] \times [4, 20] \times [1, 7]$ , and  $t \in [1, 2]$  is on a 200-point equidistant time-grid. Similar to Example 1, we used  $M = N = 500$ ,  $n = 30, 50$  and  $n_0 = \lceil n/2 \rceil, \lceil n/4 \rceil$ . The prediction accuracy measured in mean- $\log(\text{NMSPE})$  from the test set over 60 replications is shown in Figure 2.

Similar to Example 1, Figure 2 reveals that the proposed greedy algorithm outperforms its naive counterpart irrespective of the total neighborhood size ( $n$ ). As the neighborhood size gets large, the prediction accuracy of the local approximation algorithms improves, and similar to Example 1, mlegp gives better prediction accuracy than laGP for fGP implementation. In contrast to Example 1, here, fGP with mlegp implementation appears to result in the most accurate prediction (outperforming all local GPs).



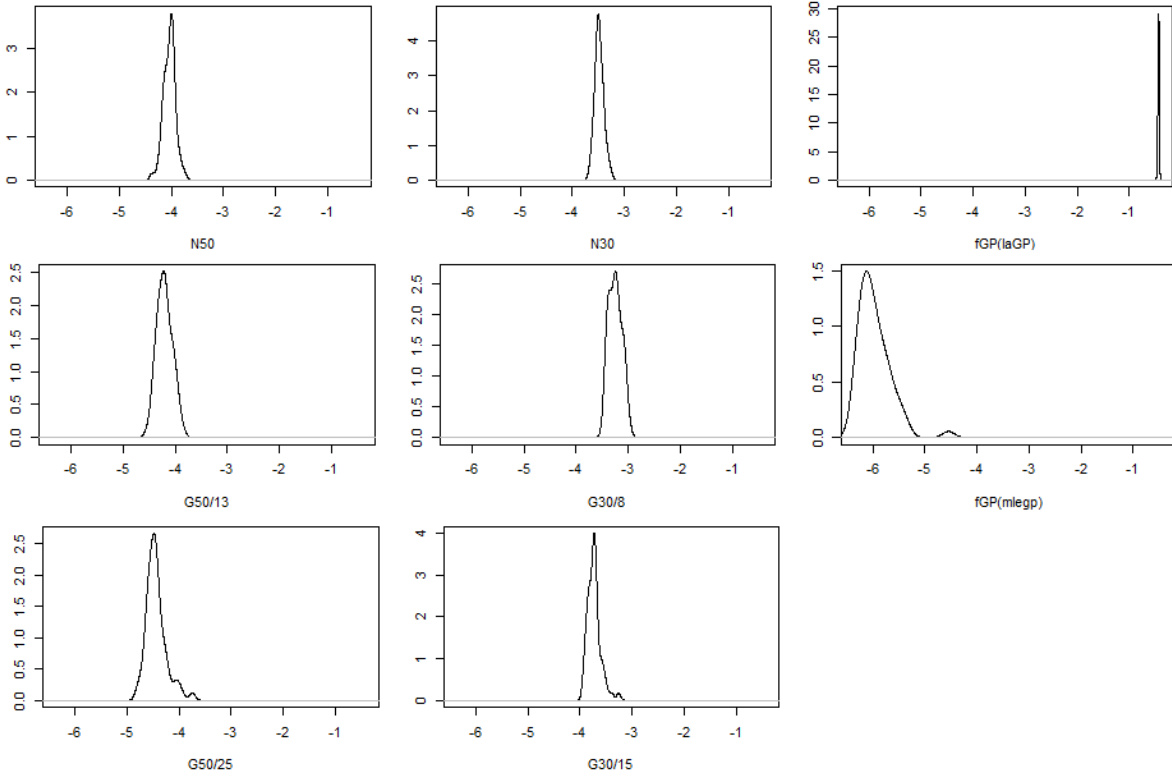


Figure 2: Density plots of the mean  $\log(\text{NMSPE})$  computed from 500 test points over 60 simulations for the computer simulator (14). The leftmost panel uses  $n = 50$ , the middle one uses  $n = 30$  and the rightmost column represents the full 500-point design.

#### 4.3. EXAMPLE 3 (Bliznyuk et al., 2008)

Consider the environmental model in Bliznyuk et al. (2008) which models a pollutant spill caused by a chemical accident. The simulator output is given by

$$f(\mathbf{x}, t) = \frac{M}{\sqrt{Dt}} \exp\left(\frac{-s^2}{4Dt}\right) + \frac{M}{\sqrt{D(t-\tau)}} \exp\left(-\frac{(s-L)^2}{4D(t-\tau)}\right) I(\tau < t), \quad (15)$$

where  $\mathbf{x} = (M, D, L, \tau, s)^T$ ,  $M$  denotes the mass of pollutant spilled at each location,  $D$  is diffusion rate in the channel,  $L$  is location of the second spill, and  $\tau$  is time of the second spill,  $\mathbf{x} \in [7, 13] \times [0.02, 0.12] \times [0.01, 3] \times [30.01, 30.295] \times [0, 3]$ ,  $t \in [0.3, 60]$  is on a regular 200-point equidistant time grid.

In this example, we increase the data size to  $N = 10,000$  training points and  $M = 3000$  test points obtained using a random LHD (McKay et al., 1979). We consider the relative performance comparison of the local SVD-based GPs, as the full GP (fGP) would be

computationally too expensive to implement on a regular office computer. To further save the computational cost, we only focus on the initial neighborhood size  $n_0 = \lceil n/2 \rceil$  for glGP, as Examples 1 and 2 demonstrate  $Gn/n_0$  to be more accurate for  $n_0 = \lceil n/2 \rceil$ . Figure 3 summarizes the distribution of 60 mean-log(NMSPE) values computed over the test set.

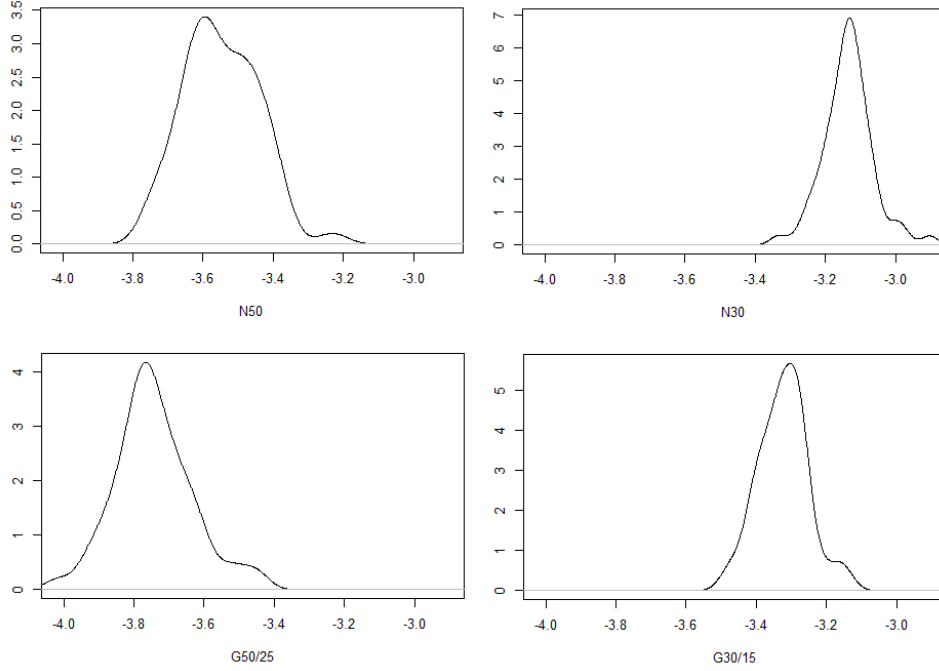


Figure 3: Density plots of the mean log(NMSPE) computed from 3000 test points over 60 simulations with inputs and outputs generated from the model (15) using a 10,000-point random LHD. The left panel corresponds to  $n = 50$ , and right one uses  $n = 30$ .

As expected, glGP outperforms nlGP, and  $n = 50$  exhibits more accurate prediction than  $n = 30$ . Since we do not have the full GP implementation, it is unclear about how good the achieved prediction accuracy is with respect to the maximum achievable. To investigate this further, we compared the prediction accuracy of  $Gn/n_0$  (with  $n_0 = \lceil n/2 \rceil$ ) for several values of  $n$ . Figure 4 summarizes the findings.

Figure 4 shows the expected increasing trend of the average prediction accuracy. Though the production accuracy increases with  $n$ , the rate of increment in the accuracy slows down as  $n$  increases, and recall that, fitting a glGP model,  $Gn/n_0$ , requires  $O(n^3NM)$  flops, which depends heavily on  $n$ .

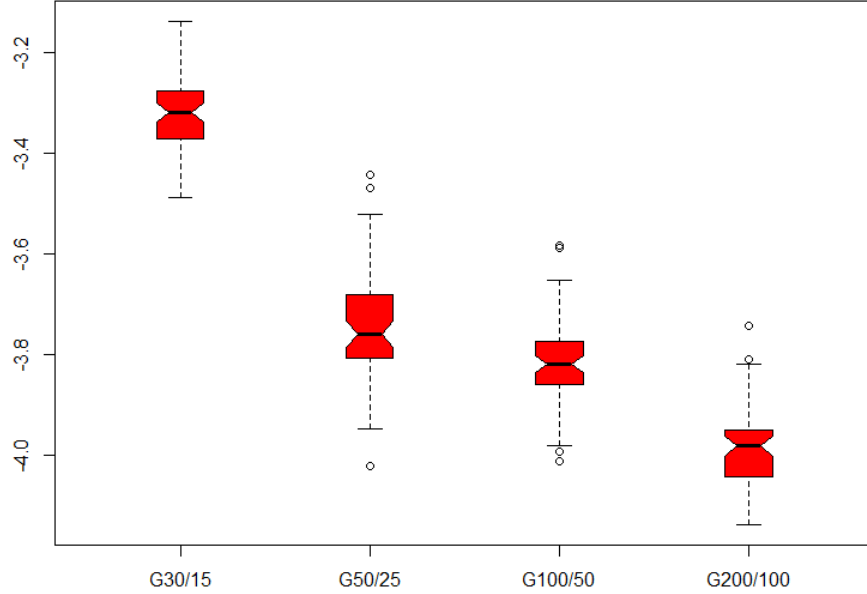


Figure 4: Boxplots of the mean  $\log(\text{NMSPE})$  computed from 3000 test points over 60 simulations for model (15) using a 10,000-point random LHD-based training data with  $n = 30, 50, 100, 200$  and  $n_0 = n/2$ .

#### 4.4. EXAMPLE 4 (Liu and West, 2009)

In this example, we consider a real-life application - a hydrological saturated path model (LogSPMs) which simulates water balance dynamics at catchment scales. These models were previously examined by Liu and West (2009) in which 8 input variables,  $N = 60$  training data points, and  $M = 60$  test data points were used. The model outputs of this experiment are time series of length  $L = 1827$  for each input. We compare the performance of the TVAR model (proposed by (Liu and West, 2009)) with the three emulators, fGP, nlGP and glGP. In addition to the prediction accuracy, we compare the execution time. Since  $N$  is small, the numerical stability in fitting fGP using the laGP package was not a concern, and the results are similar to those by the mlegp package which is thus omitted.

We implemented the TVAR model, Matlab code provided in the published supplementary material (Liu and West, 2009), on Matlab 2012a, and the three SVD-based GP models

on R-3.2.2 (R Core Team, 2015). All computation was conducted on an HP Pavilion desktop (AMD A10-6700, 3.70GHz, 8G RAM, 64-bit Windows 10). The mean NMSPEs and execution time for all methods are summarized in Table 2. Since it is a real-life application, we cannot replicate the data for a simulation study.

Table 2: Median NMSPE values over the test set and the execution time (in seconds) for TVAR, and the three SVD-based GP models emulating the response of LogSPM model.

Method	Median NMSPE	Time (s)	Method	Median NMSPE	Time (s)
TVAR	0.7158	11935.27	fGP	0.1064	1.027
N20	0.2602	3.675	N40	0.1440	13.247
G20/5	0.3634	35.097	G40/10	0.1862	177.077
G20/10	0.2802	31.592	G40/20	0.1692	151.877

Table 2 shows that the TVAR model is substantially more time consuming, and interestingly, the least accurate as compared to SVD-based GP models. Furthermore, given the computational time requirement by TVAR, one might perhaps be reluctant to use it (in its current format) for large data sets.

As expected, the full GP model (fGP) results in the most accurate prediction. More interestingly, the fGP model implementation (for this example) takes significantly less time than other local models. This is attributed to the fact that  $N = 60$  is very small, and fitting a full GP is not computationally heavy as compared to fitting a GP with 20 points. For instance, in an  $N20$  model, 60 GPs with 20 training points each have to be fit as compared to one GP with 60 points in the fGP model. That is, for such a small example, the proposed local GP model is perhaps an overkill and the full GP model wins by a big margin.

#### 4.5. Example 5 (TDB simulator - Teismann et al. (2009))

The two-delay blowfly (TDB) model (Teismann et al., 2009) simulates European red mites (ERM) population dynamics under predator-prey interactions in apple orchards via numerically solving the Nicholson’s blowfly differential equation (Gurney et al., 1980). Unmanaged ERM population growth could incur massive infestation which inflicts heavy loss in

apple industry. Therefore, the monitoring and subsequent intervention of ERM population dynamics is of vital importance for apple orchards management. The objective here is to emulate this simulator for deeper insight in the process.

The TDB model takes eleven input variables (e.g., death rates for different stages, fecundity, hatching time, survival rates, and so on) and returns the time series of ERM population evolutions at three stages, i.e., eggs, juveniles and adults (see Ranjan et al. (2016) for details). In this paper, we focus on the population dynamics of only juveniles. Figure 5 shows the output ERM juvenile population dynamics at five input points.

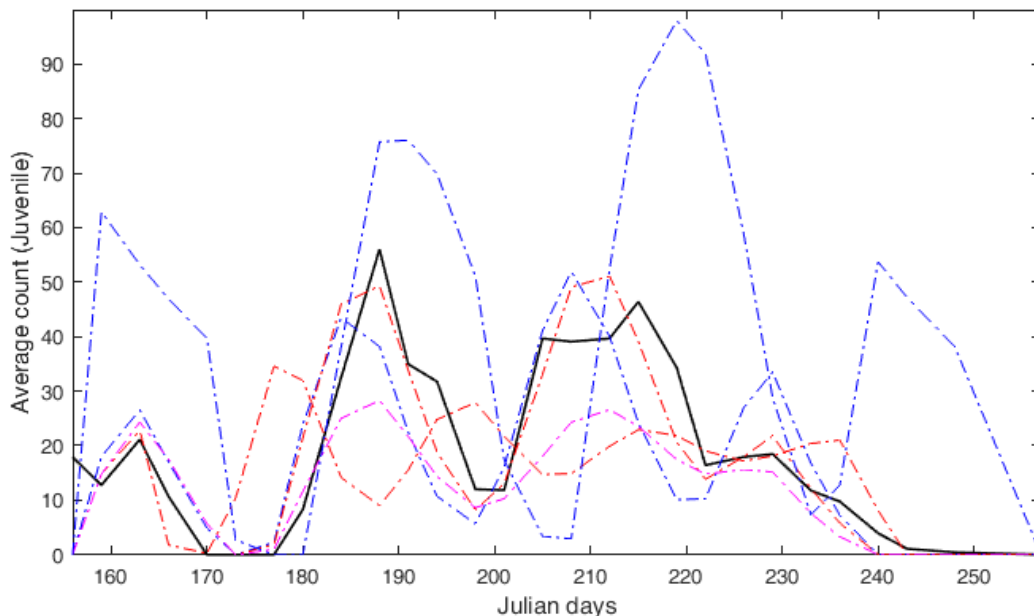


Figure 5: Juvenile ERM population dynamics as outputs of the TDB model at five different inputs. Black solid curve shows the field data and the other curves show the TDB outputs.

The input variable domains are decided by expert knowledge. For convenience, we transform the inputs into the 11-dimensional unit hypercube, and take  $y_t^*(\mathbf{x}) = \log(y_t(\mathbf{x}) + 1)$ . We take  $N = 20,000$  training and  $M = 20,000$  test points generated using maximin LHDs for the emulation of this process. For such a large scale dynamic computer model, fGP is computationally infeasible. We used  $n = 50$  and  $n_0 = \lceil n/2 \rceil$  for our local SVD-based GP models. The median NMSPEs for nlGP and glGP are 0.09034 and 0.04491 with inter-quantile differences at 0.1351 and 0.09174, respectively.

For a closer view of the prediction performance of the proposed method, we randomly select 1% of the test points, i.e., 200 points, and plot standardized prediction errors (see Figure 6). The standardized error is defined as  $(y_t^*(\mathbf{x}) - \hat{y}_t^*(\mathbf{x})) / \hat{\sigma}_t(\mathbf{x})$ , where  $\hat{\sigma}_t(\mathbf{x})$  is the predictive standard deviation of the transformed response at  $(\mathbf{x}, t)$ .

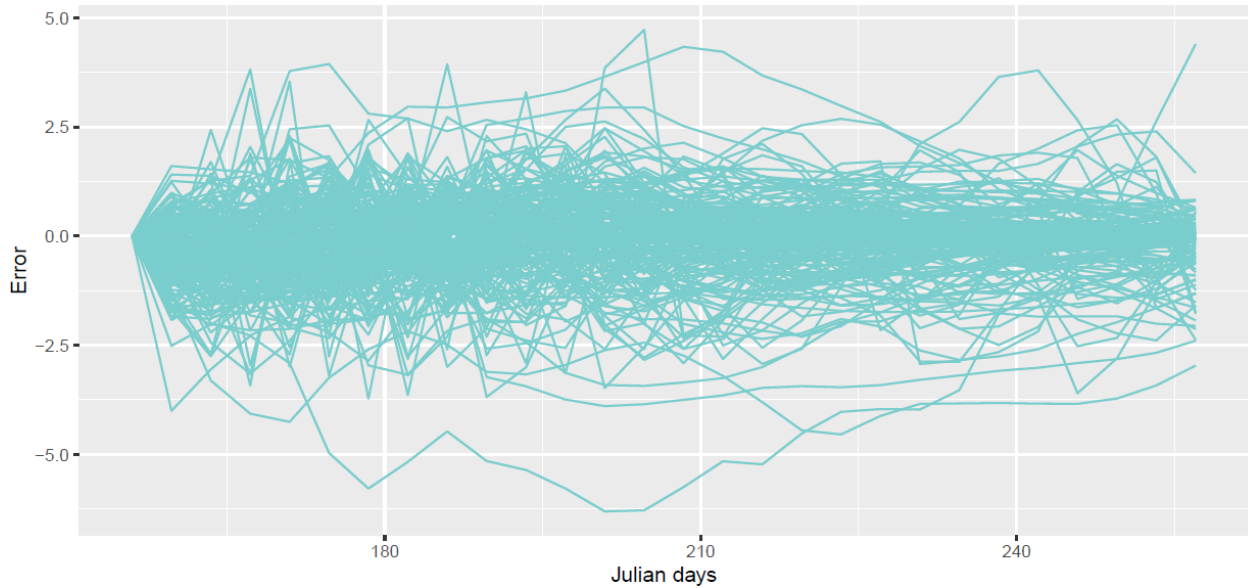


Figure 6: Standardized prediction error of  $y_t^*(\mathbf{x})$  against time (Julian days) on randomly selected 1% test points for the proposed glGP method.

As shown in Figure 6, most of the standardized error lines are concentrated within the  $\pm 2.5$  boundary, which indicates a reasonable predictive performance and uncertainty quantification.

## 5. CONCLUDING REMARKS

We have proposed local approximate SVD-based GP models for large-scale dynamic computer experiments. The proposed local SVD-based GP models with the greedy neighborhood selection algorithm reduce the time complexity of the full SVD-based GP models. Though slightly more time consuming than its naive counterpart, glGP has been shown to be much more accurate in prediction for both simulation examples and the real data analysis. With the assistance of parallel computation, the proposed algorithm can easily handle dynamic computer experiments with training set as large as 20,000 points, which is

beyond the capacity of the full model.

There are a couple of remarks worth mentioning. First, the formula (9) does not consider the possible update of the estimated correlation parameters. If new data arrives, the empirical Bayesian estimators of correlation parameters  $\theta$ 's in (7) are expected to change with the training set. To address this issue, Gramacy and Apley (2015) suggested the second order Taylor polynomial approximation. However, this approximation is only applicable for GP models with isotropic correlation functions. When we implemented this approximation approach for anisotropic Gaussian correlation functions in our experiments, it often comes across the problem of singularity (non-positive definite Hessian matrix). Nugget based methods can be used to address this issue (Ranjan et al. (2011)). Second, for successful application of glGP, we need sufficiently large neighborhood to build reliable singular vector basis. Third, there is a possible improvement in terms of computational efficiency. Instead of searching the neighborhood set for each individual point in the prediction set  $\mathbf{X}^*$ , some clustering algorithms could be performed on  $\mathbf{X}^*$  to divide it into groups on which the greedy neighborhood selection is executed.

## REFERENCES

- Bayarri, M., J. Berger, J. Cafeo, G. Garcia-Donato, F. Liu, J. Palomo, R. Parthasarathy, R. Paulo, J. Sacks, and D. Walsh (2007). Computer model validation with functional output. *The Annals of Statistics* 35(5), 1874–1906.
- Bingham, D., P. Ranjan, and W. J. Welch (2014). Statistics in action: A canadian outlook. *Sequential design of computer experiments for optimization, estimating contours, and related objectives*, 109–124.
- Bliznyuk, N., D. Ruppert, C. Shoemaker, R. Regis, S. Wild, and P. Mugunthan (2008). Bayesian calibration and uncertainty analysis for computationally expensive models using optimization and radial basis function approximation. *Journal of Computational and Graphical Statistics* 17(2), 270–294.
- Cohn, D. A., Z. Ghahramani, and M. I. Jordan (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research* 4, 129–145.

- Conti, S., J. P. Gosling, J. E. Oakley, and A. O’Hagan (2009). Gaussian process emulation of dynamic computer codes. *Biometrika* 96(3), 663–676.
- Forrester, A., A. Sobester, and A. Keane (2008). *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons.
- Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin (2014). *Bayesian data analysis*, Volume 2. Chapman & Hall/CRC Boca Raton, FL, USA.
- Gentle, J. E. (2007). *Matrix algebra: theory, computations, and applications in statistics*. Springer Science & Business Media: New York.
- Gramacy, R. B. (2005). *Bayesian treed Gaussian process models*. Ph. D. thesis, University of California Santa Cruz.
- Gramacy, R. B. (2015). *laGP: Local approximate Gaussian process regression*. R package version 1.2-1.
- Gramacy, R. B. and D. W. Apley (2015). Local gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics* 24(2), 561–578.
- Gramacy, R. B. and H. K. Lee (2012). Cases for the nugget in modeling computer experiments. *Statistics and Computing* 22(3), 713–722.
- Gurney, W., S. Blythe, and R. Nisbet (1980). Nicholson’s blowflies revisited. *Nature* 287, 17–21.
- Hager, W. W. (1989). Updating the inverse of a matrix. *SIAM review* 31(2), 221–239.
- Higdon, D., J. Gattiker, B. Williams, and M. Rightley (2008). Computer model calibration using high-dimensional output. *Journal of the American Statistical Association* 103(482), 570–583.
- Hung, Y., V. R. Joseph, and S. N. Melkote (2015). Analysis of computer experiments with functional response. *Technometrics* 57(1), 35–44.



- Jolliffe, I. (2002). *Principal component analysis*. Wiley Online Library.
- Jones, D. R., M. Schonlau, and W. J. Welch (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* 13(4), 455–492.
- Kennedy, M. C. and A. O’Hagan (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63(3), 425–464.
- Liu, F. and M. West (2009). A dynamic modelling strategy for bayesian computer model emulation. *Bayesian Analysis* 4(2), 393–411.
- Mathai, A. M. and S. B. Provost (1992). *Quadratic forms in random variables: theory and applications*. M. Dekker New York.
- McKay, M. D., R. J. Beckman, and W. J. Conover (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 42(1), 55–61.
- Morris, M. D. and T. J. Mitchell (1995). Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference* 43(3), 381–402.
- R Core Team (2015). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing.
- Ranjan, P., D. Bingham, and G. Michailidis (2012). Sequential experiment design for contour estimation from complex computer codes. *Technometrics* 50, 527–541.
- Ranjan, P., R. Haynes, and R. Karsten (2011). A computationally stable approach to gaussian process interpolation of deterministic computer simulation data. *Technometrics* 53(4), 366–378.
- Ranjan, P., M. Thomas, H. Teismann, and S. Mukhoti (2016). Inverse problem for a time-series valued computer simulator via scalarization. *Open Journal of Statistics* 6(03), 528–544.
- Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian processes for machine learning*. The MIT Press.

- Sacks, J., W. J. Welch, T. J. Mitchell, and H. P. Wynn (1989). Design and analysis of computer experiments. *Statistical Science* 4(4), 409–423.
- Santner, T. J., B. J. Williams, and W. I. Notz (2003). *The design and analysis of computer experiments*. Springer Science & Business Media: New York.
- Stein, M. L. (2005). Space–time covariance functions. *Journal of the American Statistical Association* 100(469), 310–321.
- Stein, M. L., Z. Chi, and L. J. Welty (2004). Approximating likelihoods for large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 66(2), 275–296.
- Teismann, H., R. Karsten, R. Hammond, J. Hardman, and J. Franklin (2009). On the possibility of counter-productive intervention: the population mean for blowflies models can be an increasing function of the death rate. *Journal of Biological Systems* 17(04), 739–757.

## APPENDIX: PROOF OF PROPOSITION 1

Following (6), the inner expectation in (9) can be written as

$$\begin{aligned}
& \mathbb{E} \left[ \left\| \mathbf{y}(\mathbf{x}_0) - \hat{\mathbf{y}}(\mathbf{x}_0 | \mathbf{c}(\mathbf{x}), \mathbf{V}^{*(k)}, \hat{\boldsymbol{\Theta}}^{(k)}) \right\|^2 \middle| \mathbf{c}(\mathbf{x}), \mathbf{V}^{*(k)}, \hat{\boldsymbol{\Theta}}^{(k)}, (\hat{\sigma}^{(k)})^2 \right] \\
&= \text{tr} \left( \mathbf{B}^{(k)} \boldsymbol{\Lambda}(\mathbf{V}^{*(k)}(\mathbf{x}), \hat{\boldsymbol{\Theta}}^{(k)}) (\mathbf{B}^{(k)})^T + (\hat{\sigma}^{(k)})^2 \mathbf{I}_L \right) \\
&= (\hat{\sigma}^{(k)})^2 L + \text{tr} \left( \boldsymbol{\Lambda}(\mathbf{V}^{*(k)}(\mathbf{x}), \hat{\boldsymbol{\Theta}}^{(k)}) (\mathbf{B}^{(k)})^T \mathbf{B}^{(k)} \right) \\
&= (\hat{\sigma}^{(k)})^2 L + \sum_{i=1}^{p_k} (d_i^{(k)})^2 \hat{\sigma}_i^2(\mathbf{x}_0 | \mathbf{v}_i^{(k)}(\mathbf{x}), \hat{\boldsymbol{\theta}}_i^{(k)}),
\end{aligned} \tag{A.16}$$

where  $\mathbf{V}^{*(k)}(\mathbf{x}) = [(\mathbf{V}^{*(k)})^T, \mathbf{c}(\mathbf{x})]^T$  and  $d_i^{(k)}$  is the  $i$ th largest singular value of  $\mathbf{Y}^{(k)}$ ,

$$\boldsymbol{\Lambda}(\mathbf{V}^{*(k)}(\mathbf{x}), \hat{\boldsymbol{\Theta}}^{(k)}) = \text{diag} \left( \hat{\sigma}_1^2(\mathbf{x}_0 | \mathbf{v}_1^{(k)}(\mathbf{x}), \hat{\boldsymbol{\theta}}_1^{(k)}), \dots, \hat{\sigma}_{p_k}^2(\mathbf{x}_0 | \mathbf{v}_{p_k}^{(k)}(\mathbf{x}), \hat{\boldsymbol{\theta}}_{p_k}^{(k)}) \right),$$

and

$$\hat{\sigma}_i^2(\mathbf{x}_0 | \mathbf{v}_i^{(k)}(\mathbf{x}), \hat{\boldsymbol{\theta}}_i^{(k)}) = \frac{\rho_i^{(k)}(\mathbf{x}_0, \mathbf{x})}{\alpha_i + k} (\beta_i + \psi_i^{(k)}(\mathbf{x})),$$

for  $i = 1, \dots, p_k$ , where

$$\psi_i^{(k)}(\mathbf{x}) = \mathbf{v}_i^{(k)}(\mathbf{x})^T \tilde{\mathbf{K}}_i^{-1}(\mathbf{x}) \mathbf{v}_i^{(k)}(\mathbf{x}),$$

and  $\mathbf{v}_i^{(k)}(\mathbf{x}) = [(\mathbf{v}_i^{(k)})^T, c_i(\mathbf{x})]^T$  is the  $i$ th column of  $\mathbf{V}^{*(k)}(\mathbf{x})$ .

The first equality of (A.16) follows from Theorem 3.2b.1 of Mathai and Provost (1992). The third equality is derived from the column-orthogonality of  $\mathbf{B}^{(k)}$ , i.e.  $(\mathbf{B}^{(k)})^T \mathbf{B}^{(k)} = (\mathbf{D}^{*(k)})^2$ .

Plugging (A.16) into (9), we get

$$\begin{aligned} J(\mathbf{x}_0, \mathbf{x}) &= \mathbb{E} \left[ (\hat{\sigma}^{(k)})^2 L + \sum_{i=1}^{p_k} (d_i^{(k)})^2 \hat{\sigma}_i^2(\mathbf{x}_0 | \mathbf{v}_i^{(k)}(\mathbf{x}), \hat{\boldsymbol{\theta}}_i^{(k)}) \middle| \mathbf{V}^{*(k)}, \hat{\boldsymbol{\Theta}}^{(k)}, (\hat{\sigma}^{(k)})^2 \right] \\ &= (\hat{\sigma}^{(k)})^2 L + \sum_{i=1}^{p_k} (d_i^{(k)})^2 \left( \frac{\rho_i^{(k)}(\mathbf{x}_0, \mathbf{x})}{\alpha_i + k} (\beta_i + \mathbb{E}[\psi_i^{(k)}(\mathbf{x}) | \mathbf{V}^{*(k)}, \hat{\boldsymbol{\Theta}}^{(k)}, (\hat{\sigma}^{(k)})^2]) \right) \\ &= (\hat{\sigma}^{(k)})^2 L + \sum_{i=1}^{p_k} (d_i^{(k)})^2 \left( \frac{\rho_i^{(k)}(\mathbf{x}_0, \mathbf{x})}{\alpha_i + k} (\beta_i + \mathbb{E}[\psi_i^{(k)}(\mathbf{x}) | \mathbf{v}_i^{(k)}, \hat{\boldsymbol{\theta}}_i^{(k)}]) \right) \\ &= (\hat{\sigma}^{(k)})^2 L + \sum_{i=1}^{p_k} (d_i^{(k)})^2 \left( \frac{\rho_i^{(k)}(\mathbf{x}_0, \mathbf{x})}{\alpha_i + k} \left( \beta_i + \frac{\alpha_i + k}{\alpha_i + k - 1} \psi_i^{(k)} \right) \right). \end{aligned}$$

The second equality holds because  $\rho_i^{(k)}(\mathbf{x}_0, \mathbf{x})$  is a deterministic function of  $\mathbf{x}_0, \mathbf{x}$  and  $\hat{\boldsymbol{\theta}}_i^{(k)}$ . The third equality follows from the independence among  $c_i$ 's. The validity of the fourth equality is due to Gramacy and Apley (2015).